The integration of OntoClean in WebODE

Mariano Fernández-López, Asunción Gómez-Pérez

Facultad de Informática . Universidad Politécnica de Madrid
Campus de Montegancedo, s/n. 28660 Boadilla del Monte. Madrid. Spain
{mfernandez, asun}@fi.upm.es

Abstract. Enterprises will only be interested in the use of ontologies if such ontologies are evaluated enough. Therefore, the development of ontology evaluation tools is a crucial matter. We have built the ODEClean module in the workbench for building ontologies named WebODE. ODEClean allows cleaning taxonomies following the OntoClean method, and WebODE provides technical support to the Methontology methodology for building ontologies. We approached the development of this module in two steps. Firstly, we have integrated the OntoClean method into the conceptualisation activity of Methontology. Secondly, we have designed and implemented ODEClean using a declarative approach for specifying the knowledge to be used on the evaluation. ODEClean uses: (a) the Top Level of Universals, (b) meta-properties based on philosophical notions, and (c) OntoClean evaluation axioms. The main advantage of this approach is that the system could easily allow the user relax or stress the evaluation of the taxonomy just selecting more or less meta-properties.

1 Introduction

Currently the semantic web [1] attracts researchers from all around the world. Numerous tools and applications of semantic web technologies are already available [2] [3] [4] and the number is growing fast [5]. Ontologies play an important role for the semantic web as a source of formally defined terms for communication. They aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused, shared, and operationalised across applications and groups. The large visibility of the semantic web, its tools and applications already attracts industrial partners, e.g. in numerous projects funded by the European Commission. As they move from academic institutions into commercial environments they have to fulfil stronger requirements (e.g. concerning correctness, consistency, completeness, conciseness, etc.). Therefore, the evaluation is a key activity in ontology development. Some of the most well-known proposals for ontology evaluation are: Gómez-Pérez's proposal [14] [15] [16], Kalfoglou and colleagues' proposal [22][23], and OntoClean [26]. OntoClean is a method for cleaning tangled taxonomies founded in philosophical notions as: rigidity, identity, unity, etc.

Most of the methodologies and methods ([25], [18], [10], [24], etc) for building ontologies include an evaluation activity. Most of the times, ontology evaluation is done once the ontology is finished and implemented in a given ontology language. Methontology [12] proposes to evaluate the ontology during its whole life cycle: it

recommends to carry out most of the evaluation of the content at the conceptualisation activity to prevent the detection of faults in the ontology code. WebODE [8] is the workbench that gives technological support to some activities of Methontology.

However, Methontology does not propose a set of design principles that guide the development of taxonomic knowledge and methods to clean tangled taxonomies. Therefore, given that OntoClean allows cleaning wrong *subclass of* links in taxonomies using notions like *rigidity*, *identity* and *unity*, it is an appropriate complement to be used for building taxonomies at the conceptualisation activity in Methontology. As a consequence, we integrated OntoClean method in Methontology, as we presented in [11].

Once the unification at the methodological level was performed, we were in the appropriate situation to build the software that gives support to OntoClean in WebODE. We call this software ODEClean. The inclusion of the ODEClean module would allow the use of the OntoClean method in an efficient way. Indeed, until now, OntoClean is being used in several industrial and academic settings to evaluate taxonomies [19]. However it is usually applied by hand.

Our solution also fits with the idea presented in [13], since we have not built an isolated tool for OntoClean, but a module integrated in an ontological engineering workbench.

The base of ODEClean is Guarino and their colleagues' top-level level ontology of universals [20], whose instances are concepts (in contrast with the top-level of particulars, whose instances are individuals). We have implemented the top level ontology of universals in WebODE. We enriched this ontology including the **meta-properties** (rigidity, identity, unity) and the evaluation rules proposed by OntoClean method. Then, the ontology was completly translated automatically using WebODE translators into Ciao Prolog. Thus, ODEClean consults the enriched top-level of universals and the axioms in Prolog every time that it has to evaluate a given domain ontology. That is, **the main advantage of the ODEClean module is that the knowledge used to evaluate ontologies is declaratively expressed through an ontology inside our ODEClean module. The user could relax or to stress the evaluation just clicking on more or less meta-properties**

Section 2 will present OntoClean, section 3 will present the top-level ontology of universals, section 4 will show ODEClean's ontology (enriched top-level of universals), section 5 will present WebODE, section 6 will show the ODEClean plug-in, its functions and how we have developed it, and, finally, section 7 will be devoted to conclusions and future lines.

2 OntoClean method

OntoClean has been elaborated by the Ontology Group of the LADSEB-CNR in Padova (Italy). It is a method to clean taxonomies according to notions such as: *rigidity*, *identity* and *unity*. Let us see these notions [17]:

- *Rigidity*. This notion is defined based on the idea of essence. A property is essential to an individual if and only if necessarily holds for that individual. Thus, a property is *rigid* (+R) if and only if is necessarily essential to all its instances. A property is *non-rigid* (-R) if and only if it is not essential to some of its instances, and *anti-rigid* (~R) if and only if it is not essential to all its instances. For

example, the concept `person` is usually considered rigid, since every person is essentially such, while the concept `student` is not normally considered anti-rigid, since every student can possibly be a non-student a few years later.

- *Identity*. A property *carries an identity criterion* (IC) (+I) if and only if all its instances can be (re)identified by means of a suitable "sameness" relation. A property *supplies an identity criterion* (+O) if and only if such criterion is not inherited by any subsuming property. For example, *person* is usually considered a supplier of an identity criterion (for example the fingerprint), while `student` just inherits the identity criterion of `person`, without supplying any further identity criteria.
- *Dependency*. An individual *x* is constantly dependent on *y* if and only if, at any time, *x* cannot be present unless *y* is fully present, and *y* is not part of *x*. For example, a hole in a wall is constantly dependent on the wall. The hole cannot be present if the wall is not present. A property *P* is constantly dependent if and only if, for all its instances, there exists something they are constantly dependent on. For instance, the concept `hole` is constantly dependent because every instance of `hole` is constantly dependent.
- *Unity*. We can say that an individual is a *whole* if and only if it is made by a set of parts unified by a relation *R*. For example, the enterprise Iberia is a whole because it is composed by a set of people that are linked by the relation `having the same president`. A property *P* is said to *carry unity* (+U) if there is a *common* unifying relation R such that all the instances of *P* are wholes under R^1. For example, the concept `enterprise-with-president` carries unity because every enterprise with president is made up people linked through the relation `having the same president`. A property carries *anti-unity* (~U) if all its instances can possibly be non-wholes. Properties that refer to amounts of matter, like *gold*, *water*, etc., are good examples of anti-unity.

Note that the definition of these notions refer to properties of properties. For example, *rigid* is a property that can take different values in different properties (*yes* in `person`, *no* in `student`, etc.). Another example is *carries an identity criterion*, since it can also take different values in different properties (*yes* in `person`, *no* in `student`, etc.). These properties of properties are called **meta-properties**, and to indicate their values, special symbols are used. For example, +R means that the meta-property *rigid* has the value *yes*. The meta-properties are useful to detect wrong *subclass of* relations. For example, `person` cannot be subclass of `student` because the former one is rigid and the later one not. In fact, if we had this link, what would it happen if a person was not student any more?

According to LADSEB-CNR's proposal, the specific steps to clean the wrong *subclass of* links in a taxonomy are (based on [26] and interviews with LADSEB-CNR's group):

1) *Put tags to every property assigning meta-properties*. This eases the analysis, because all the meta-properties are simultaneously visible.

---

[1] In the actual definition, the authors use *essential wholes* instead of *wholes*. We will sometimes sacrifice the accuracy to make clear the ideas of this paper to people still non very familiarised with Formal Ontology.

2) *Focus just on the rigid properties*. A taxonomy without rigid properties is called *backbone taxonomy*. It is the base of the rest of the taxonomy, that is, the essential part.
3) *Evaluate the taxonomy taking into account principles based on the meta-properties*. For instance, a rule suggested in OntoClean is "a property carrying anti-unity has to be disjoint of a property carrying unity". As a consequence, "a property carrying unity cannot be a subclass of a property carrying anti-unity". Therefore, `bronze statue` (it carries unity) cannot be a subclass of `bronze` (it carries anti-unity), for example.
4) *Consider non-rigid properties*. When the backbone taxonomy has been examined, the modeller has to evaluate the non-rigid properties. One of the proposed rules is: "a rigid property and an anti-rigid property are ever disjoint". As a consequence, "a non anti-rigid property cannot be a subclass of an anti-rigid property". Therefore, `person` (rigid) cannot be a subclass of `student` (anti-rigid).
5) *Complete the taxonomy with other concepts and relations.* There can be several reasons to introduce new concepts. One of them is the transformation of concepts in relations, for example, `student` could be transformed into a relation between `person` and `university`.

OntoClean has been used by IBM, OntologyWorks[2], Document Development Corporation[3]. At the Italian National Research Council Laboratories (LADSEB-CNR and ITBM-CNR), in Padova and Rome, OntoClean is in use in several projects including the development of an upper-level ontology based on a restructuring of WordNet, and the development of a core ontology for financial knowledge interchange [19].

3 Top level Ontoloy of Universals

The LADSEB-CNR's Ontology Group (in Italy) has built two top-level ontologies, as presented in figure 1: one of universals, and another of particulars. Universals are concepts like `car` or `computer`, etc. and individuals are instances of these concepts, like `my car` or `my computer`, etc. Thus, for example, the particular `my car` is an instance of the universal `car`.

Top Level Ontology of Universals (TPU) is made up by meta-concepts like `type` or `role`, for example (see figure 2) [20]. The instances of such meta-concepts are concepts (universals). Concerning Top Level Ontology of Particulars (and every domain ontology) it is made up by concepts (universals) whose instances are particulars. That is, the tag "universals" or "particulars" associated to the names of the two CNR's ontologies are given by the kind of instances that they can contain.

[2] www.ontologyworks.com
[3] www.docdev.com

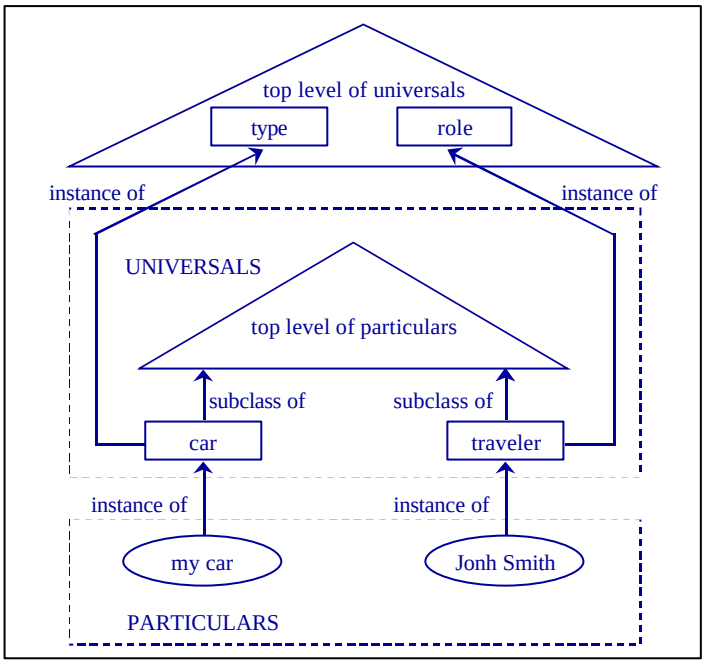

Fig. 1. Relationship between particulars and universals

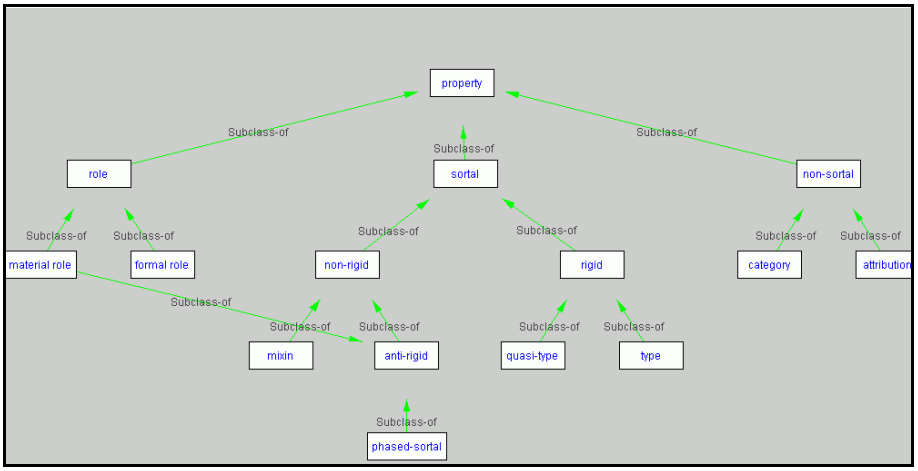

Fig. 2. Class taxonomy of the top-level ontology of universals

4 ODEClean's ontology

LADSEB-CNR's group continues its research in defining well-defined criteria for cleaning taxonomies, therefore, the proposed axioms can be modified and extended.

That is, every tool that implements OntoClean should be flexible. Consequently, we have taken a declarative approach to implement the knowledge used to clean taxonomies in ODEClean. Moreover, the representation of OntoClean rules to clean taxonomies also requires the representation of knowledge about meta-properties (*rigid*, *carries an identity criterion*, etc.). Because of this, ODEClean uses the top-level ontology of universals [19] enriched with LADSEB-CNR's meta-properties [17] and evaluation axioms [26].

To build ODEClean's ontology in WebODE, we mixed the following components:

1) *The top level of universals*. We introduced the taxonomy that appears in figure 2, which was obtained from [20].
2) *Meta-properties*. They were introduced as instance attributes of the root of TPU (`property`) according to WebODE knowledge model. Figure 3 shows the meta-concept `property` and its attributes.
3) *OntoClean axioms*. The OntoClean axioms to evaluate ontologies that appear in [26] were also included in TPU using the WebODE WAB module. Figure 4 shows the axiom that says : "a non anti-rigid property cannot be a subclass of an anti-rigid property".

During its working, ODEClean automatically links every concept inserted in the ontology into the root of its ontology through the relation *instance of*, as we can see in figure 3. Consequently, the TPU ontology meta-properties will be meta-attributes (class attributes) of every concept of the ontology to be cleaned. Hence, the user can assign values to the meta-properties in every concept of the ontology that (s)he is building.

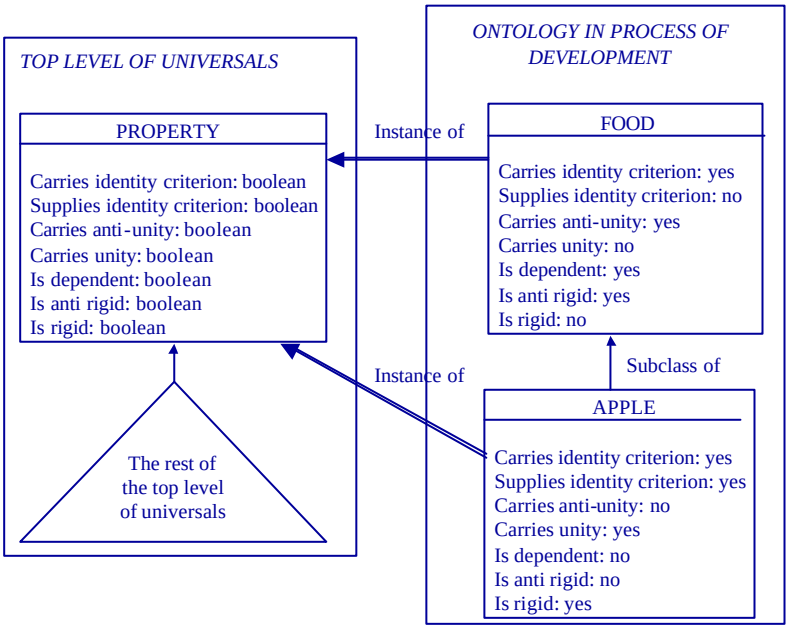

Fig. 3. Links between the top-level of universals and the ontology in process of development

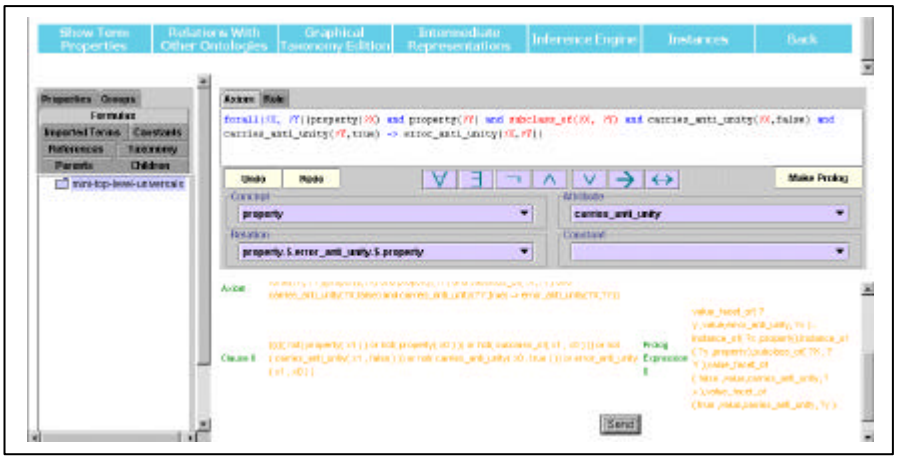

Fig. 4. OntoClean axiom in WebODE

In the current version of ODEClean the complete TPU hierarchy is not necessary, since OntoClean meta-properties are defined in a single meta-concept. However, the complete TPU hierarchy will be very useful. On the one hand, the values that the meta-properties take in the domain ontologies could be used to automatically classify the domain ontology concepts as instances of the meta-concepts of the TPU ontology (`role`, `type`, etc.). In fact, each TPU meta-concept has values associated to different meta-properties. For example, every role is anti-rigid, dependent, etc. On the other hand, TPU is already a part of OntoClean [26]. When the ontologist has to assign meta-property values to a domain concept, (s)he can take into account if that domain concept (for example, `food`) is a role, a type, etc. Indeed, some meta-properties values in the domain concepts could be deduced from the links between the domain ontology and TPU.

Nowadays, the problem to use TPU as a part of OntoClean is to know which meta-concept is each domain ontology concept instance of. Even more, depending on the point of view adopted by the modeller, the same concept can be, for example, a role or a type. In any case, ODEClean already includes the different meta-property values through all its ontology. Thus, for example, the meta-property *anti-rigid* takes the value *yes* in the meta-concept *role*. In this way, ODEClean is already prepared to help, in the future, in meta-property value inference.

5 WebODE

WebODE is a scalable, integrated workbench for ontological engineering that eases the representation of ontologies, the reasoning with ontologies and the exchange of ontologies with other ontology tools and ontology-based applications [8]. It has been developed by the Ontology Group of the Technical University of Madrid. The WebODE's knowledge model [6] is based on the intermediate representations proposed in Methontology [10]. Hence, it allows modelling concepts and their attributes (both class and instance attributes), taxonomies of concepts, disjoint and

exhaustive class partitions, ad-hoc binary relations between concepts, properties of relations, constants, axioms and instances of concepts and relations.

WebODE is built according to a four-tier architecture: client, presentation, business logic, and database tiers. In all these tiers, we have used standard technology. The client tier uses HTML, XML, CSS, JavaScript and Java applets. The presentation tier uses servlets and JSPs. The business logic tier uses Java and RMI-IIOP. Finally, the database tier uses JDBC and Oracle. The main WebODE services are:

- *The WebODE Ontology Editor.* It allows the collaborative construction of ontologies at the knowledge level. It provides a default form-based web user interface to create ontologies according to the knowledge model aforementioned. The WebODE Ontology Editor also includes *OntoDesigner*, a visual tool that aids in the construction of concept taxonomies and ad-hoc relations between concepts.
- *WebODE Axiom Builder (WAB)*. WAB is an axiom and rule editor that is integrated in the WebODE Ontology Editor. It allows creating first order logic axioms and rules using a graphical user interface. It also provides a library of built-in axioms, which can be reused for creating other axioms, rather than building them from scratch.
- *WebODE's inference engine service*. WebODE includes an OKBC-based inference engine. This inference engine reasons with a subset of the OKBC protocol's primitives [7].
- *WebODE interoperability services*. Ontologies built with WebODE can be easily integrated in other ontology servers or used in ontology-based applications. Possible choices for interoperability include WebODE's ontology access API, which can be accessed by other applications using RMI, and is completely compliant with the WebODE's knowledge model. Currently, WebODE is able to export to and import ontologies from: RDF(S), OIL, DAML + OIL, the XMLization of CARIN and FLogic. It also can export to JESS and Prolog.
- *WebPicker* [9] is a set of wrappers that allow importing standards of classification of products and services in the context of electronic commerce into WebODE (UNSPSC, e-cl@ss and RosettaNet). We are currently extending it to wrap other sources of information, such as Cyc.
- *ODECatalogue* is able to generate electronic catalogs from ontologies according to some parameters. The catalogue generation from an ontology assures a correct and rich classification of the different products.
- *ODEMerge* performs a supervised merge of concepts, attributes and relationships from two different ontologies built for the same domain, according to semantic criteria and resources used for natural language processing.
- *ODEClean* plug-in, which will be presented in this paper.

WebODE has been successfully used, with different domains and purposes and by different groups of people, in the following projects: The European IST project MKBEEM (IST 1999-10589), the OntoWeb thematic network (IST-2000-29243), the Spanish CICYT project ContentWeb (TIC-2001-2745), the Spanish CICYT project on Methodology for Knowledge Management (TIC-980741), etc.

6 The ODEClean plug-in of WebODE

To present the plug-in, first of all, we show its functions (section 6.1), and then, we will describe how ODEClean module has been built (section 6.2). In section 6.2 we will not describe the integration process of OntoClean in METHONTOLOGY because it was presented at [11].

6.1 Functions of the ODEClean plug-in of WebODE

The purpose of ODEClean is to allow developers to evaluate taxonomies using OntoClean method. ODEClean is a plug-in of WebODE and WebODE was designed taking into account the METHONTOLOGY methodology. When the ontologists build an ontology in WebODE, it is possible for him to select wheather he wants to build the taxonomy taking into account the OntoClean principles. It is also possible to pick up an ontology from WebODE ontology library and to clean its taxonomy just assigning values of the meta-properties of each concepts. One way to assign meta-properties to the concepts is through the form-based web user interface of WebODE (see figure 5). The other way to assign meta-properties is through the visual tool *OntoDesigner* (see figure 6). This last way allows the developer to tag the concepts of the ontology like if (s)he was designing the taxonomy in a blackboard.

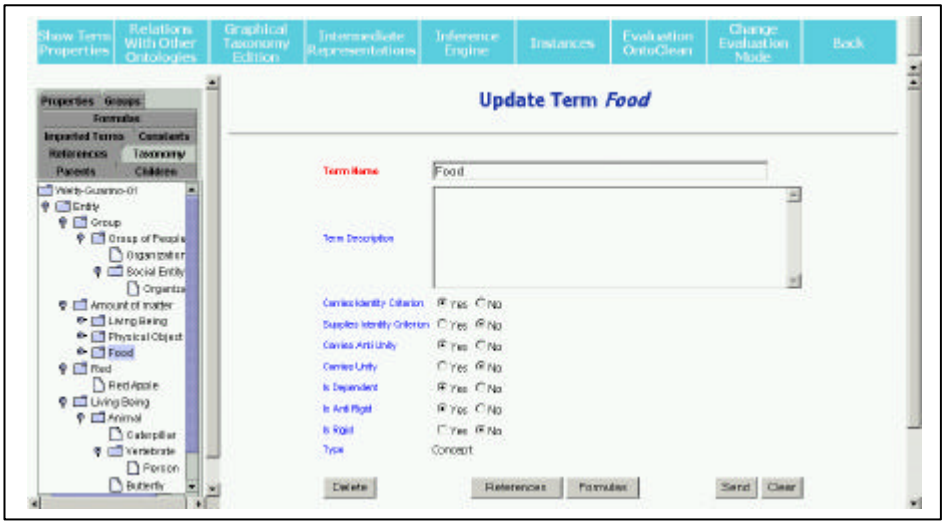

Fig. 5. Form-based web for ODEClean

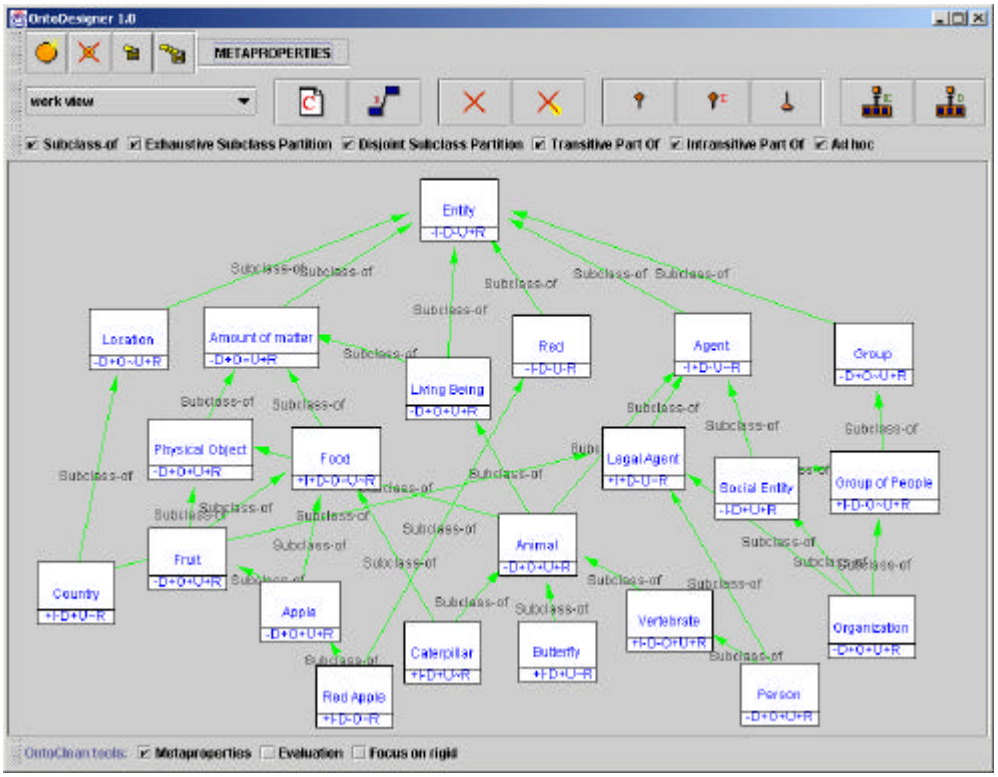

Fig. 6. OntoDesigner for evaluating taxonomies following OntoClean (taxonomy taken from [26], where the authors use it to show how to evaluate ontologies with OntoClean)

The main functions provided by ODEClean are:

1. *Establishing the evaluation mode*. The user can choose whether the system has to show the errors every time that it detects a problem in the domain ontology, or the system only has to show the errors when the user ask for them. This option is available in the button *Change Evaluation Mode* of the form-based web (see top figure 5), whereas it is available in the signal *Evaluation* (figure 6) of OntoDesigner.
2. *Assigning meta-properties to concepts*. The user will be able to set up meta-properties concerning identity, unity, dependency and rigidity. If the form-based web is used, then a change in the value of a meta-property can provoke an automatic change in the value of other meta-property. For example, if you click in *yes* in *supplies an identity criterion*, then the value of *carries an identity criterion* is automatically established as *yes*. On the other hand, the assignment of values to the meta-properties using the *OntoDesigner* is performed tagging each concept with the OntoClean classical symbols introduced in section 2 (*~R+I-O*, etc.). A user that does not wish to see the meta-properties with OntoDesigner can

hide them clicking in *Metaproperties*.

3. *Focusing on rigid properties*. The user can decide whether to show or not the non-rigid properties. As you can see in section 2, one of the step of OntoClean is to focus on rigid properties.
4. *Evaluation according to the taxonomic constraints*. If the user order to evaluate the ontology, then the found errors are shown. Each error message describes the violation of a OntoClean axiom (see [6]) in a link *subclass of* between two concepts. The first error that appears in figure 7, for example, shows that the concept `food` is anti-rigid whereas `apple` (a subclass of `food`) does not. This is a violation of OntoClean axioms.

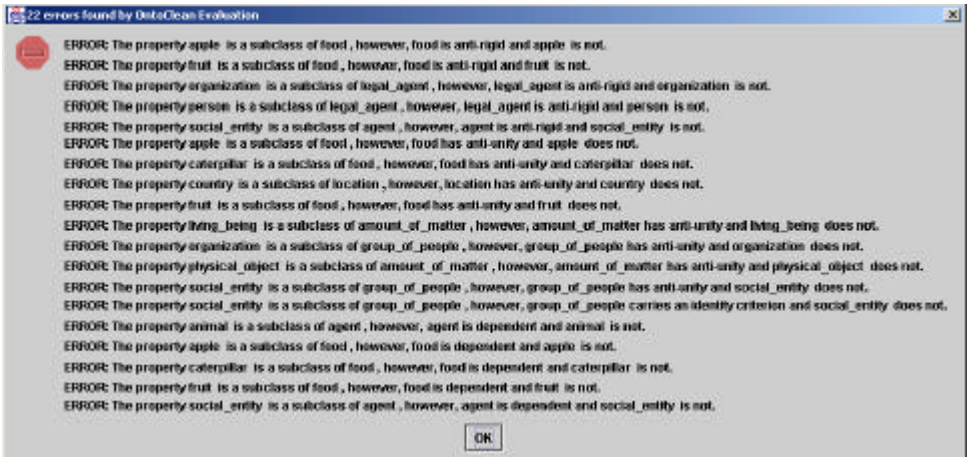

Fig. 7. Errors detected by ODEClean

6.2 How we have built ODEClean

To develop ODEClean, we firstly built TPU using the WebODE Ontology Editor. We enriched it with the necessary meta-properties for OntoClean. Then, using WAB, we added the LADSEB-CNR's rules into the top level of universals. Then, we translated this ODEClean's ontology into Prolog using the WAB service of WebODE. Such Prolog ontology is the base of our system.

Thus, the particular steps that we have carried out to develop ODEClean are (see figure 8):

1. *ODEClean's ontology building*. As we have said in section 4, we made an ontology that contains OntoClean knowledge, useful for taxonomy cleaning.
2. *Translating into Prolog of ODEClean's ontology*. The purpose of this step was to generate a code with inference engine available. We used the WebODE translator that generates Prolog. WebODE translator into Prolog uses OKBC primitives. The use of OKBC primitives could ease the interaction with other systems.

3. *Building the rest of the system.* Taking the Prolog ontology, we built the rest of the modules of ODEClean: the user interface and the communication with the rest of WebODE.

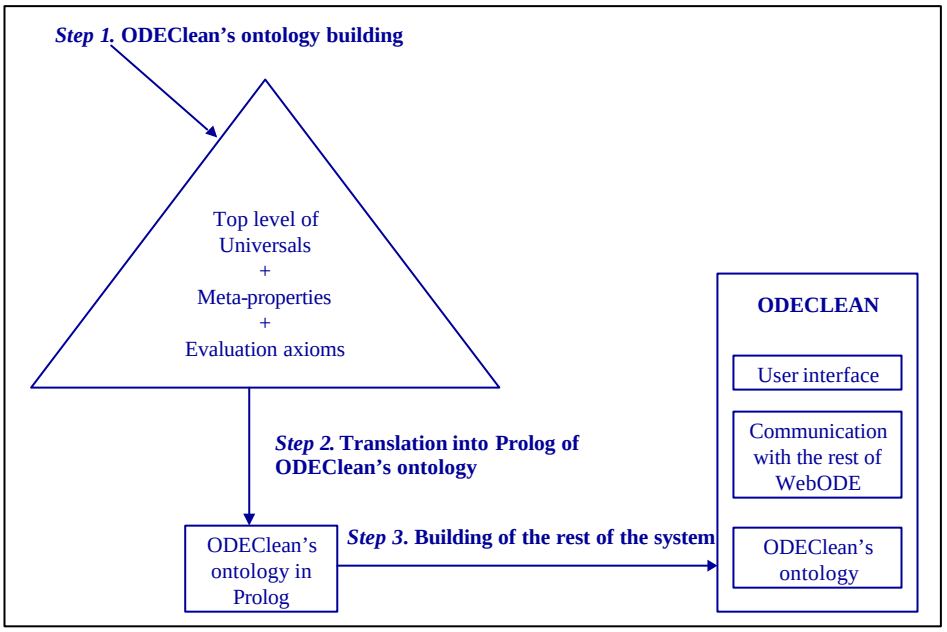

Fig. 8. The development of ODEClean

Concerning the internal behaviour of the system, WebODE's inference engine makes use of Ciao Prolog [21]), as a consequence, the inference engine that applies the OntoClean rules uses Ciao Prolog.

7 Conclusions

In this paper we have presented the plug-in of WebODE that implements OntoClean, the method to clean ontologies elaborated in the LADSEB-CNR of Padua (Italy). WebODE is the ontology development platform developed by the Ontology Group of the Technical University of Madrid. This plug-in allows the developer to assign meta-properties to concepts, focus on non-rigid properties, automatically check errors, etc. The user can visualise the ontolology either through a form-based web user interface or graphically with OntoDesigner.

This plug-in is not only the product of software development, but also a work at the ontology development methodological level. That is, first of all, we integrated OntoClean in METHONTOLOGY. Then, we made the ODEClean plug-in integrated in WebODE, the METHONTOLOGY software support.

The plug-in has been built using as base LADSEB-CNR's top-level ontology of universals translated to Prolog. We have used the WebODE WAB plug-in to add it the OntoClean evaluation rules before translating it to Prolog.

The main contributions of our work are:

1. The new module is a consequence of the integration of an evaluation method in a development methodology. That is, we have carried out an integration at the methodological level before performing it at the software level.
2. We have built the first tool integrated in a ontology development platform that supports the method OntoClean.
3. An ontology built by a group that has not participated in the development of WebODE has been introduced in WebODE. Moreover, the ontology enriched with meta-properties and axioms coded in Prolog is thought to be reusable in other platforms or tools different to WebODE.

   Kalfoglou and colleagues' evaluation of applications is also based on the use of ontologies. However, their approach is more focussed on the use of an ontology as the formal a specification of the application that they are going to evaluate.
4. **The knowledge used to evaluate ontologies is declaratively specified**,. which means that:
   - New meta-properties could be added easily, just introducing new attributes in ODEClean's ontology.
   - New axioms could be added or modified using WAB.

According to our experience developing this plug-in, if the future evaluation tools are declaratively developed, they will be flexible.

Acknowledgements

This work is supported by the project "ContentWeb: Plataforma tecnológica para la web semántica: ontologías, lenguaje natural y comercio electrónico[4]" (TIC-2001-2745), and by the project "Esperonto Services" (IST-2001-34373). This work would not have been possible without the help of Emilio Raya.

References

1. T. Berners-Lee, J. Hendler and O. Lassila. A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2002, cf. http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html.
2. EU IST-1999-10132 project “On-To-Knowledge: Content-driven knowledge management tools through evolving ontologies”, cf. http://www.ontoknowledge.org.
3. US DARPA project “DARPA Agent Markup Language (DAML)”, cf. http://www.daml.org.
4. EU IST-2000-29243 thematic network “OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce”, cf. http://www.ontoweb.org.

[4] ContentWeb: Platform for the Semantic Web: ontologies, natural language and e-commerce

5. A survey on ontology tools. Deliverable D13. IST OntoWeb Thematic Network. May 2002.
6. Arpírez, J.C.; Corcho, O.; Fernández-López, M.; Gómez-Pérez, A. WebODE: a scalable ontological engineering workbench. First International Conference on Knowledge Capture (K-CAP 2001). Victoria, Canada. October, 2001.
7. Chaudhri V. K.; Farquhar A.; Fikes R.; Karp P. D.; Rice J. P. The Generic Frame Protocol 2.0. Technical Report, Stanford University.1997.
8. Corcho, O., Fernández-López, M., Gómez-Pérez, A., Vicente, O. WebODE: an integrated workbench for ontology representation, reasoning and exchange. 13th International Conference on Knowledge Engineering and Knowledge Management EKAW02. 2002.
9. Corcho, O; Gómez-Pérez, A. WebPicker: Knowledge Extraction from Web Resources. 6th Intl. Workshop on Applications of Natural Language for Information Systems (NLDB'01). Madrid. June, 2001.
10. Fernández-López, M.; Gómez-Pérez, A. "Overview and analysis of methodologies for building ontologies". *Knowledge Engineering Representation* (to be published).
11. Fernández-López, M.; Gómez-Pérez, A.; Guarino, N. 2001. "The Methontology & OntoClean merge". *Technical Report, OntoWeb special interest group on Enterprise-standards Ontology Environments*. Amsterdam. 2001.
12. Fernández-López, M.; Gómez-Pérez, A.; Pazos, J.; Pazos, A. Building a Chemical Ontology using methontology and the Ontology Design Environment. IEEE Intelligent Systems and their applications. #4 (1):37-45. 1999.
13. Gómez-Pérez, A. *A proposal of infrastructural needs on the framework of the semantic web for ontology construction and use*. FP6 Programme Consultation Meeting 9. April 27th, 2001.
14. Gómez-Pérez, A. *Evaluation of Ontologies*. International Journal of Intelligent Systems. 16(3). March, 2001.
15. Gómez-Pérez, A. Some ideas and Examples to Evaluate Ontologies. Technical Report KSL-94-65. Knowledge System Laboratory. Stanford University. Also in Proceedings of the 11th Conference on Artificial Intelligence for Applications. CAIA94. 1994.
16. Gómez-Pérez, A. From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment. Technical Report. KSL-94-73. Knowledge Systems Laboratory. Stanford University. December 1994.
17. Gangemi, A., Guarino, N., Masolo, C., and Oltramari, A. 2001. Understanding top-level ontological distinctions. Proc. of IJCAI 2001 workshop on Ontologies and Information Sharing .
18. Grüninger, M.; Fox, M. S. 1995. "Methodology for the design and evaluation of ontologies." *Workshop on Basic Ontological Issues in Knowledge Sharing*. Montreal (Canada).
19. Guarino, N. and Welty, C. 2002. "Evaluating Ontological Decisions with OntoClean". *Communications of the ACM*, 45(2): 61-65.
20. Guarino, N. and Welty, C. 2000. A Formal Ontology of Properties. In R. Dieng and O. Corby (eds.), Knowledge Engineering and Knowledge Management: Methods, Models and Tools. 12th International Conference, EKAW2000. Springer Verlag: 97-112.
21. Hermenegildo, M., Bueno, F., Cabeza, D., Carro, M., García, M., López, P., Puebla, G. *The Ciao Logic Programming Environment*. International Conference on Computational Logic (CL2000). July, 2000.
22. Y.Kalfoglou, D.Robertson. "Managing Ontological Constraints", In Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, August 1999.
23. Y.Kalfoglou, D.Robertson,"Use of Formal Ontologies to Support Error Checking in Specifications" In Proceedings of the 11th European Workshop on Knowledge Acquisition, Modelling and Management (EKAW99), Dagsthul, Germany, May 1999.

24. Staab, S.; Schnurr, H.-P.; Studer, R.; Sure; Y. “Knowledge Processes and Ontologies”, *IEEE Intelligent Systems, 16(1)*, January/February 2001.
25. Uschold, M. King, M. 1995. “Towards a Methodology for Building Ontologies”. *Workshop held in conjunction with IJCAI on Basic Ontological Issues in Knowledge Sharing*.
26. Welty, C.; Guarino, N. *Supporting Ontological Analysis of Taxonomic Relationships*. Data and Knowledge Engineering. September 2001.